Claim 1. A method for causing scheduler software to produce code which executes rapidly including the steps of:

reordering a sequence of instructions to run as fast as possible even though the reordered sequence may generate an exception,

raising an exception if the reordered sequence of instructions violates a scheduling constraint, and

determining steps to be taken for correctly executing each set of instructions about which an exception is raised.

Claim 2. A method as claimed in claim 1 in which the step of raising an exception if the reordered sequence of instructions violates a scheduling constraint includes a step of detecting exceptions caused by reordered instructions in the reordered sequence.

Claim 3. A method as claimed in claim 2 in which the step of detecting exceptions caused by reordered instructions in the reordered sequence includes;

remembering an instruction which has been placed out of order in the sequence, and

checking instructions in the sequence with respect to which the remembered instruction has been reordered to determine if an incorrect result is produced by the sequence of instructions.

Claim 4. A method as claimed in claim 2 in which the step of remembering an instruction which has been placed out of order in the sequence includes storing a memory address accessed by the instruction, and

the step of checking instructions in the sequence with respect to which the remembered instruction has been reordered to determine if an incorrect result is produced by the sequence of instructions includes comparing the stored memory address with memory addresses accessed by instructions against which it is checked.

Claim 5. A method as claimed in claim 4 in which the memory address of the data accessed by the instruction is stored in a protection register.

Claim 6. A method as claimed in claim 4 including a further step of remembering where all remembered instructions are held.

Claim 7. A method as claimed in claim 1 in which it is probable that the reordered sequence will generate an exception.

Claim 8. A method as claimed in claim 7 in which the step of raising an exception if the reordered sequence of instructions violates a scheduling constraint includes a step of detecting exceptions caused by reordered instructions in the reordered sequence.

Claim 9. A method as claimed in claim 8 in which the step of detecting exceptions caused by reordered instructions in the reordered sequence includes;

remembering an instruction which has been placed out of order in the sequence, and

checking instructions in the sequence with respect to which the remembered instruction has been reordered to determine if an incorrect result is produced by the sequence of instructions.

Claim 10. A method as claimed in claim 8 in which the step of remembering an instruction which has been placed out of order in the sequence includes storing a memory address accessed by the instruction, and

the step of checking instructions in the sequence with respect to which the remembered instruction has been reordered to determine if an incorrect result is produced by the sequence of instructions includes comparing the stored memory address with memory addresses accessed by instructions against which it is checked.

Claim 11. A method as claimed in claim 10 in which the memory address accessed by the instruction is stored in a protection register.

Claim 12. A method as claimed in claim 10 including a further step of remembering where all remembered instructions are held.

Claim 13. A system for controlling the reordering of instructions to produce code which executes rapidly comprising:

means for reordering a sequence of instructions to run as fast as possible even though the reordered sequence may generate an exception,

means for raising an exception if the reordered sequence of instructions violates a scheduling constraint, and

means determining steps to be taken for correctly executing each set of instructions about which an exception is raised.

Claim 14. A system as claimed in claim 13 in which the means for raising an exception if the reordered sequence of instructions violates a scheduling constraint includes means for detecting exceptions caused by reordered instructions in the reordered sequence.

Claim 15. A system as claimed in claim 14 in which the means for detecting exceptions caused by reordered instructions in the reordered sequence comprises:

means for identifying an instruction which has been placed out of order in the sequence, and

means for checking instructions in the sequence with respect to which the identified instruction has been reordered to determine if an incorrect result is produced by the sequence of instructions.

Claim 16. A system as claimed in claim 14 in which the means for identifying an instruction which has been placed out of order in the sequence includes means for storing a memory address accessed by the instruction, and

the means for checking instructions in the sequence with respect to which the identified instruction has been reordered to determine if an

incorrect result is produced by the sequence of instructions includes means for comparing the stored memory address with memory addresses accessed by instructions against which it is checked.

Claim 17. A system as claimed in claim 16 in which the means for storing the memory address of the data accessed by the instruction is a protection register.

Claim 18. A system as claimed in claim 16 further comprising means for storing indicators for the memory addresses accessed by identified instructions.

Claim 19. A system as claimed in claim 16 further comprising means for storing indicators for the memory addresses accessed by identified instructions which exist during a sequence of instructions, and

means for storing indicators for the addresses accessed by identified instructions which persist beyond a sequence of instructions.

Claim 20. A system as claimed in claim 19 in which the means for storing indicators for the memory addresses accessed by identified instructions which exist during a sequence of instructions is a first register storing indications of valid memory addresses accessed by identified instructions,

the means for storing indicators for the memory addresses accessed by identified instructions which persist a sequence of instructions is a second register storing indications of valid memory addresses accessed by identified instructions; and

further including a third register for storing indicators for the memory addresses accessed by identified instructions, and

means for transferring indications from the second register to the first and third registers when a sequence of instructions is executed without an exception, and

means for transferring indications from the third register to the first and second registers when an exception is generated during a sequence of instructions being executed.

Claim 21. A system as claimed in Claim 13 further comprising means for replicating memory data in an execution unit register.

ADD A17
Add B1